SYSTEM AND METHOD FOR OLAP REPORT GENERATION WITH SPREADSHEET REPORT WITHIN THE NETWORK USER INTERFACE

Related Applications

This application is related by subject matter to co-pending applications entitled "System and Method for Asynchronous Control of Report Generation Using a Network Interface," Attorney Docket No. 53470.000005 and "System and Method For Network User Interface OLAP Report Formatting," Attorney Docket No. 53470.000012, both of which are filed herewith.

Field of the Invention

This invention relates to a system and method for presenting reports generated by an on-line analytical processing system using a spreadsheet application within a network user interface.

Background of the Invention

The ability to act quickly and decisively in today's increasingly competitive marketplace is critical to the success of any organization. The volume of data that is available to organizations is rapidly increasing and frequently overwhelming. The availability of large volumes of data presents various challenges. One challenge is to avoid inundating an individual with unnecessary information. Another challenge it to ensure all relevant information is available in a timely manner.

One known approach to addressing these and other challenges is known as data warehousing. Data warehouses and data marts are becoming important elements of many information delivery systems because they provide a central location where a

reconciled version of data extracted from a wide variety of operational systems may be stored. As used herein, a data warehouse should be understood to be an informational database that stores shareable data from one or more operational databases of record, such as one or more transaction-based database systems. A data warehouse typically allows users to tap into a business's vast store of operational data to track and respond to business trends that facilitate forecasting and planning efforts. A data mart may be considered to be a type of data warehouse that focuses on a particular business segment.

Decision support systems have been developed to efficiently retrieve selected information from data warehouses. One type of decision support system is known as an on-line analytical processing system ("OLAP"). In general, OLAP systems analyze the data from a number of different perspectives and support complex analyses against large input data sets.

There are at least three different types of OLAP architectures - ROLAP, MOLAP, and HOLAP. ROLAP ("Relational On-Line Analytical Processing") systems are systems that use a dynamic server connected to a relational database system. Multidimensional OLAP ("MOLAP") utilizes a proprietary multidimensional database ("MDDB") to provide OLAP analyses. The main premise of this architecture is that data must be stored multidimensionally to be viewed multidimensionally. A HOLAP ("Hybrid On-Line Analytical Processing") system is a hybrid of these two.

ROLAP is a multi-tier, client/server architecture comprising a presentation tier, an application logic tier and a relational database tier. The relational database tier

stores data and connects to the application logic tier. The application logic tier comprises a ROLAP engine that executes multidimensional reports from multiple end users. The ROLAP engine integrates with a variety of presentation layers, through which users perform OLAP analyses. The presentation layers enable users to provide requests to the ROLAP engine. The premise of ROLAP is that OLAP capabilities are best provided directly against a relational database, *e.g.*, the data warehouse.

In a ROLAP system, data from transaction-processing systems is loaded into a defined data model in the data warehouse. Database routines are run to aggregate the data, if required by the data model. Indices are then created to optimize query access times. End users submit multidimensional analyses to the ROLAP engine, which then dynamically transforms the requests into SQL execution plans. The SQL is submitted to the relational database for processing, the relational query results are cross-tabulated, and a multidimensional result set is returned to the end user. ROLAP is a fully dynamic architecture capable of utilizing precalculated results when they are available, or dynamically generating results from atomic information when necessary.

The ROLAP architecture directly accesses data from data warehouses, and therefore supports optimization techniques to meet batch window requirements and to provide fast response times. These optimization techniques typically include application-level table partitioning, aggregate inferencing, denormalization of data structures, multiple fact table joins, and the use of specific RDB optimizer tactics which vary with each particular brand of relational database.

MOLAP is a two-tier, client/server architecture. In this architecture, the MDDB serves as both the database layer and the application logic layer. In the

database layer, the MDDB system is responsible for all data storage, access, and retrieval processes. In the application logic layer, the MDDB is responsible for the execution of all OLAP requests. The presentation layer integrates with the application logic layer and provides an interface through which the end users view and request OLAP analyses. The client/server architecture allows multiple users to access the multidimensional database.

Information from a variety of transaction-processing systems is loaded into the MDDB System through a series of batch routines. Once this atomic data has been loaded into the MDDB, the general approach is to perform a series of batch calculations to aggregate along the orthogonal dimensions and fill the MDDB array structures. For example, revenue figures for all of the stores in a state would be added together to fill the state level cells in the database. After the array structure in the database has been filled, indices are created and hashing algorithms are used to improve query access times.

Once this compilation process has been completed, the MDDB is ready for use. Users request OLAP reports through the presentation layer, and the application logic layer of the MDDB retrieves the stored data.

The MOLAP architecture is a compilation-intensive architecture. It principally reads the precompiled data, and has limited capabilities to dynamically create aggregations or to calculate business metrics that have not been precalculated and stored.

The hybrid OLAP ("HOLAP") solution is a mix of MOLAP and relational architectures that support inquiries against summary and transaction data in an

integrated fashion. The HOLAP approach enables a user to perform multidimensional analysis on data in the MDDB. However, if the user reaches the bottom of the multidimensional hierarchy and requires more detailed data, the HOLAP engine generates an SQL statement to retrieve the detailed data from the source relational database management system ("RDBMS") and returns it to the end user. HOLAP implementations rely on simple SQL statements to pull large quantities of data into the mid-tier, multidimensional engine for processing. This constrains the range of inquiry and returns large, unrefined result sets that can overwhelm networks with limited bandwidth.

As described above, each of these types of OLAP systems are typically client-server systems. The OLAP engine resides on the server side and a module is typically provided at a client-side to enable users to input queries and report requests to the OLAP engine. Current client-side modules are stand alone software modules that are loaded on client-side computer systems. One drawback of such systems is that a user must learn how to operate the client-side software module in order to initiate queries and generate reports.

Although various user interfaces have been developed to enable users to access the content of data warehouses through server systems, many such systems experience significant drawbacks. One system enables the user to access information using a network interface device (*e.g.*, a web browser). Such systems provide the advantage of allowing users to access the content of a data warehouse through existing web browser applications residing on their desktop or other personal computer connected

over a network to the server system. Such systems however suffer from various drawbacks.

In existing web-based user interfaces, a user inputs a query or report request through the web browser, the web browser initiates the report to the server, and waits until the report finishes before providing results back through the web browser to a user. While the web browser is waiting for the report, the web browser "locks up," thereby requiring that the user open another browser window if another site or page is desired to be opened. Accordingly, if the delay between the report initiation and completion is lengthy, some users may incorrectly assume that the delay is caused by an error, when in reality the delay may simply be caused by the complicated nature of the query. Users may become impatient and resubmit the report by selecting the reload or refresh feature of the web browser. Current systems treat each such request separately and, therefore, each time the user resubmits a request, another query for the same information is submitted to the server. Because the reason for the delay in most circumstances is the size or complexity of the query, the server is then hit with processing multiple complex or large queries simultaneously when only one such query or report is desired. Accordingly, these multiple reports tax server resources unnecessarily.

Another drawback this example illustrates is that current systems focus on requests. Accordingly, each request is treated as a different request that is submitted to the server for processing. Accordingly, identical requests initiated by different users are submitted to the server system for processing, thus utilizing the server

resources twice to generate the same report. These and other drawbacks exist with current web-based data warehouse/server system user interfaces.

Summary of the Invention

An object of the present invention is to overcome these and other drawbacks in existing systems.

Another object of the present invention is to provide a system and method for enabling system users to present reports generated by an on-line analytical processing system using a spreadsheet application within a network user interface.

Another object of the present invention is to provide a system and method for enabling system users to personalize reports based on user preference.

These and other objects of the present invention are realized according to various embodiments of the present invention. Accordingly, an embodiment of the present invention comprises a system and method for enabling users of a multi-user network-based OLAP system to present reports generated by the OLAP system using a spreadsheet application within a network user interface. A client-side network user interface, such as a web browser, is presented to a user over the network so that the user may submit a request for a report. The server-side system receives the request and processes the report. When the server-side system completes the report processing, the report is communicated to the client-side network user interface. The report may then be presented to the user(s) that submitted requests for that report. The client-side network user interface comprises a spreadsheet application that may be used to display the report to the user.

As part of the invention, the reports may include macros to be processed by the server system upon completion of the report to format the report based on the results of the report. The macros may be defined using visual basic for applications (VBA) language that is provided by the spreadsheet application. Accordingly, the functionality of the spreadsheet application is leveraged to provide a more dynamic, maintainable, scalable and manipulatable system. Further, by performing the macros at the server, the processing performed by the client-side application is reduced by avoiding performance of what may be work-intensive processing steps to format the report based on these macros. Also, the macros reduce the amount of data transmitted over the network and thereby improve the download time.

Additionally, the system provides an API to interface between the spreadsheet program macros and the report processing module. In generating a report, the components of the report are defined to be addressable features. Accordingly, through the API, various functions may be called to address the components of the report to be included in macros. This enables the user to programmatically address the data in the report.

Further, the server-side upon completion of the workbook processing and macro formatting may cache the workbook at the server-side for later access by users. For example, client-side macros may be employed that may first attempt to resolve themselves through the software application operating at the client-side. If not, however, the cached workbook may be accessed to resolve the macro to avoid having to regenerate the reports in the workbook.

One advantage of the present invention is that the system enables users to display reports using a spreadsheet application directly within a network user interface. This enables users to utilize the functionality offered by the spreadsheet application. Users may therefore apply formatting and various other operations to the report using the spreadsheet application. Additionally, this reduces the need to alternate among various applications operating on the system. A user is not required to switch between a network user interface from which a report request is transmitter and a spreadsheet application used to display the report. The report is displayed within the network user interface and enables users to access all of the features of the spreadsheet application.

Further, the present invention provides a system for generating creating personalized (both via autoprompts and personalization filters), dynamic/adhoc, scalable, optimized (uses multiple levels of caches, such as caches on the server and the network output module), secure (using the appropriate user's credentials to get only the data that user has access to), maintainable (utilizing macros that apply formatting based on the content of a report rather than hard-coding of that formatting information). Further, using the macros, a report may be reduced to key information for a particular user from the spreadsheet generated from the workbook.

Other objects and advantages of the present invention exist.

Brief Description of the Drawings

Fig. 1 is a schematic block diagram of a method for generating reports asynchronously in accordance with an embodiment of the invention.

Fig. 2 is a schematic block diagram of a method for generating, monitoring, and canceling reports asynchronously in according with another embodiment of the present invention.

Fig. 3 is a schematic illustration of a report list in accordance with one embodiment of the invention.

Fig. 4 is a schematic illustration of a status page in accordance with an embodiment of the invention.

Fig. 5 is a schematic block diagram of an asynchronous report requesting network system in accordance with an embodiment of the invention.

Fig. 6 is a schematic block diagram of system components in accordance with an embodiment of the invention.

Fig. 7 is a schematic block diagram of a network-based system in which the asynchronous report generation system may operate in accordance with one embodiment of the invention.

Detailed Description of the Preferred Embodiments

Accordingly, a system and method for presenting reports generated by an OLAP system using a spreadsheet application within a network user interface is provided. The system may comprise an Internet-based web browser interface that connects users over the Internet to a server system that connects to the OLAP system for processing requested reports.

A method 200 may be provided according to an embodiment of the present invention, as depicted in Fig. 1. In this method, a user system providing a user interface module may be connected over a network to an OLAP server system. The

first step in method 200 then involves step 202 whereby the user may request the processing of a report to be run on the OLAP system through a web browser or other user interface system. The report may then be transmitted by the web browser or other user interface system to the server system such as a network module in communication with the web server.

After a report has been requested, in step 204, the system compares that report with other pending or recently completed reports for the user submitting the report and for other users connected to the system to determine if the requested report is the same as a pending or processed report. In an embodiment, if a particular report has recently been completed or is already in the queue to be processed by the server system, the system does not create a new report to be processed. Rather, in step 207, the system adds the user to the report that has already been requested so that when the previously requested report completes, then all users that requested that report receive the results. If the report has already completed, then the system immediately returns the results to the user as part of steps 215 and 226. If the report has not been requested, then in step 208, a report request is generated and transmitted to the OLAP processing system.

In step 215, the system returns control of the network interface, such as a web browser or other user interface module, to the user so that the user may perform other tasks. Specifically, the server system may direct the web server to transmit a signal to enable the user to use the web browser to perform other tasks, including submission of another report. The web server may also transmit a page that informs the user that the report that was requested was received and is scheduled for processing, such as a status page as discussed below.

Then, in step 226, the system notifies the user when a report has been completed so that the user may review the results of the report. By returning control of the web browser to the user upon receipt of the report request, the system frees the user to submit multiple reports for processing in an asynchronous manner. Additional details and features of the present invention may be understood with respect to Figs. 2-7 below.

Fig. 2 illustrates a more detailed embodiment of method 200 in accordance with one embodiment of the invention. Specifically, in step 204, the system determines whether that report is the same, substantially the same or similar to a report that has previously been requested. To do so, the system may store or have access to all, or a certain predetermined subset of all, reports forwarded to the system within a predetermined period of time (e.g., several hours, several days, or several weeks).

In an embodiment, the system, in step 204, may compare the name of the report with the name of other reports maintained in a cache that comprise all reports already pending and reports that have been completed but not removed from the cache. Also, the system may compare the template, filter, and user view of a request to the template/filter/user view combination of other requested reports in the cache. If the template/filter/user view combination selected are identical, then the reports may be deemed to be the same.

Also, two reports may also be determined to be the same for purposes of avoiding generating a new report even if the templates or filters are not identical. For example, two reports may be deemed to be substantially the same if the template

/filter/user view combination is within a predetermined range of a template/filter combination of a previously run or pending report. For example, if a report for gross sales of running shoes for March, 1999 in the U.S. has been forwarded for processing, it may be determined that a request for gross sales of a particular brand of shoes for March, 1999 in the U.S. is substantially the same or similar enough to be the same report. Other methods of determining similarity of reports to reduce the number of jobs to be processed by the system may also be used.

In step 208, if the report is different from other pending or recently completed reports, then the report is generated by placing the report in the processing queue in a known manner. Additionally, the report may be added to a report list in step 210. The report list may comprise a list of all reports generated within a predetermined period of time and may include pending and completed reports. The report list may contain a report entry for each report containing the names or identifier(s) of system users that have requested that report. The report entry for each report may also contain additional information if desired (*e.g.*, complete request parameters, user id, requests GUID, electronic mail address, employee numbers, metadata user id, project name, report types (prompt or non-prompt), auto-prompt selections, report description, report name, filter name, template name, folder name, requested time, submitted time, completed time, filter details, status, and operators used (AND, OR, NOT)). Whether the report is new or previously requested, the request is then added to the report entry for the report in step 212. The request may identify the user names, grid/graph, user ID, time requested, etc.

It may also be desirable to notify the user of the status of the report request so in step 214, a status page may be generated. Therefore, in step 214, a status page may be generated and presented to the user by the system over the network. The status page may identify the name of the report, date and time submitted, and/or any other additional information. An example of a status page is depicted in Fig. 4. The status page may be displayed as a separate window and may be accessed by the user through viewing a report list for that user as described below. At this point, the system may enable the user to request another report, view old reports or perform other types of actions.

The status page may also enable the user to cancel a particular request, as for example, illustrated in Fig. 4 and described below. The user may also be able to cancel a report from a user report list, as for example, the user report list shown in Fig. 3, such as by clicking on the report entry and selecting a cancel icon, pull-down menu option, or other mechanism for canceling a report.

By treating each report separately and asynchronously, a report may be canceled at any time. As each report may have multiple users that have been added to that report, however, in one embodiment, a report is not canceled from the queue at the system until all users that have been added to that report have canceled. If a user cancels the report, the user's name or id is removed from the report entry for that report and that canceled user then cannot access the report results unless they resubmit another request for the report. The details of this process are depicted in steps 216-226.

Specifically, in step 216, the system determines whether a report request has been canceled. If a report request has been canceled by a user, the system removes the request for that user from the report list in step 218. After removing the user from the report entry for that report, the system, in step 220, determines whether any users remain in the report entry for that canceled report. If any user names remain, method 200 returns to step 216. If no other user names remain in the report entry, the system terminates the report and deletes it from the queue of reports being processed by the server system in step 222.

If, in step 216, a report request has not been canceled, the system determines whether any of the reports are complete in step 224. If a report is complete, in step 226 the system notifies all of the system users identified in the report list for that report that the report has been completed. Notification may be made using e-mail, facsimile, pager, through the web interface such as through the web browser or other user interface module using push technology, etc. or any other mechanism as described below.

If the report generation is not complete, in step 224 the system returns to step 216 to determine whether any report request cancellations have been entered into the system. This continues until the report generation is complete and all systems users are notified that the report generation is complete for all reports on the system. The process may involve polling whereby steps 216 and 224 are performed on a predetermined schedule, such as every seven seconds or so. Other schedules may also be used.

Method 200 provides numerous advantages for generating reports. Receiving report requests and processing those requests asynchronously enables users to operate other applications while a report is being generated. Additionally, by focusing on the identity of the report rather than a request for that report, duplicative report generation is minimized. Therefore, if several users request the same report, only one report is generated.

Fig. 3 illustrates a user-specific report list 300 in accordance with an embodiment of the invention. Report list 300 may contain a report list for reports submitted for a specific user within a predetermined period of time. That time may be a particular log-in session or a discrete time interval for example. For each report, report list 300 may contain a "name" field 302 which identifies the name of the report being generated. A folder field 303 may be provided for indicating the folder or other category that the report may be organized. A "submitted" field 304 may be used to identify the date and/or time the report request was submitted. Other fields, for example, a "details" field 306 may be used to provide additional information, such as, the status of the report. Fields 302-306 may be sorted according to various criteria (*e.g.*, ascending or descending order) by, for example, clicking on a field using a conventional computer mouse. It is to be understood that report list 300 may be sorted according to the value of a field (*e.g.*, user, date, report, etc.). A status field may also be provided and contain information as described below with respect to Fig. 4. Further, various icons or other symbols may be used to indicate the status or type of report.

As Fig. 3 depicts, various types of reports may be processed according to the present invention and these different report types may be displayed differently in report list 300. For example, an auto-prompt report may be provided with details of the prompting to show the filters, metrics, what information has been drilled on, what was selected in the prompt, other filter details, and other parameters selected for the report.

As discussed above, to notify the user that the request has been received and is being processed, a status page may be presented to the user. A status page may be used to display the progress of a report being generated and a user-specific report list may be viewed.

Fig. 4 illustrates a status page 400 in accordance with one embodiment of the invention. Status page 400 provides a view displaying the current status of a report's execution cycle. The execution cycle identifies the progress of a report generation (*e.g.*, 10% complete, Error, complete with errors, pending, complete, sending request, generating SQL, submitting job request, request in queue, running report, retrieving results, cross-tabbing data, formatting data, report complete, error running report, report canceled, m of n reports completed). Status page 400 may contain a graphical display 402 of the progress of the report generation on a percentage complete basis. Selectable icons may also be displayed within status page 400. For example, a report list icon 404 may be displayed which enables a user to display a report list. A "refresh" icon 406 may be used to manually update the progress of the report generation. As described in further detail below, this operation may be automated and

set by a user. Additionally, a "cancel request" icon may be displayed which permits a user to cancel the generation of a requested report.

A refresh status page rate may be set by a system user or by a system administrator. The system maintains a master list of requested reports and stores the list on a server. The master list may be refreshed when caches expire, a report is canceled, or a report is removed from a cache for another reason, etc.

A system administrator master report list may also be provided according to the present invention and enabled through network output module 22. The administrator report list may be similar to user report list 300 but may additionally comprise entries for each report to identify the users that have requested that report. Further, the system administrator report list may comprise all reports run within a predetermined period of time and may be broken out project by project if desired. The report list may provide a mechanism that enables a system administrator to delete entire reports, groups of reports, to delete a particular user or groups of users from a particular report or modify the order of reports in the queue. The administrator may thereby control reports submitted to the server system. In one embodiment, all such reports may be stored in a cache and the master report list may contain all cached reports plus all pending reports.

The information in the history list may include the name of the request, the report display mode (graph or grid), the date and time the request was last submitted, the status of the request, the users who have submitted the request and their identifiers or keys, filter details for report requests, and workbook report names for workbook

requests. The administrator may refresh the list, delete a request or view details about a request. The details may include the filters and templates requested.

According to another embodiment of the present invention, users may be able to submit requests for workbooks to be processed by the server system on data in the data warehouse. A workbook may comprise complex reports that generally comprise a plurality of reports and template/filter combinations that are collected to be generated as a large complex report. Therefore, as the term "report" as used herein should be understood to comprise a single template/filter combination run against data in the data warehouse or a collection of such "reports" in a format known as a "workbook."

Generally, to generate a workbook request, the user logs into the server system through username and password combinations. The user interface then presents the user with options for designing a new workbook or modifying an existing workbook. To add a new workbook or modify an existing one, the user identifies the workbook (*e.g.*, by naming it), and then selects one or more predefined reports or template/filter combinations to be included in the workbook. The user also selects the sequence of the reports and template/filter combinations. The server system may present the available reports and template/filter combinations by accessing the data warehouse, as for example, with a ROLAP engine by displaying the available table entries from the relational database structure in the data warehouse. Auto-format and drill-down reports may also be included in a workbook.

A workbook is processed as a collection of reports, with each report being processed and then when all reports in the workbook are complete, the workbook is

formulated and the users requesting the workbook are then notified. The reports in a workbook may be processed asynchronously or sequentially depending on the types of reports.

Workbooks may be implemented through leveraging spreadsheet application programs, such as Microsoft EXCEL™. Through these spreadsheet applications, formatting of multiple reports into a combined workbook may be provided. Accordingly, the programming language in the spreadsheet application may be used to configure the format and arrangement of reports within the workbook. For example, Visual Basic for Applications (VBA) may be used in the EXCEL™ spreadsheet software application. This provides flexibility in formatting and combining reports to create new ones and in building an interactive spreadsheet application for the end user. For example, a spreadsheet application for a stock market workbook may provide pull-down menus for selecting different time ranges and industries which then update instantly/dynamically. Because these programming languages and spreadsheet programs, such as VBA and EXCEL are open standards, they are widely used and therefore, the usefulness of the workbook feature is enhanced.

Also, using EXCEL™, for example, enables users to select various properties and formatting for the reports that are to be generated. Integration of EXCEL™ with the invention allows users to use all of the functionality offered in EXCEL™ and to apply that functionality to the reports generated using the invention. For example, EXCEL™ offers existing macros that may be used to automate work flow or other features for the reports. Additionally, these macros may be modified or additional macros may be recorded to perform a specific operations desired by a user. The

macros may be used, for example, to facilitate work flow. Generating reports in EXCEL™ also enables reports to display grids and graphs of the same report at the same time, display headers and footers, deliver multiple worksheets in a single workbook, support logical exceptions through use of overwriting rules, allows administrators to create formatted reports, etc.

The system enables EXCEL™ (or other spreadsheet application) integration by utilizing the browser functionality and storing information about the report in a frame to enable the report to be refreshed. The invention may use macros offered in the spreadsheet application to logically combine and manipulate reports. In this manner, reports are generated independent of X/Y coordinates of data which may change as the data changes. Because reports are presented in EXCEL™, reports may be manipulated, saved, and viewed while off-line.

Another advantage is the ability to view multiple reports at a time. Spreadsheet applications generally permit users to view more than one worksheet at a time. Therefore, by integrating a spreadsheet application with an on-line analytical processing system, this feature may also be utilized. A grid and graph of the same report may be displayed at the same time. This may be achieved through a macro that takes the data and displays it as both a grid and graph and places those within the browser of the user.

Another important feature of the present invention is the inclusion of macros within the workbook that may be processed by the server upon processing of the report. The macros may utilize the spreadsheet program to apply formatting to the report based on the contents of the report. This may be achieved by assigning

definitions to portions of a report and enabling the macro to define formats using the definitions. Therefore, rather than specifying that the third row is to be bold, the macro may specify that the heading row should be bold. The rows that are defined as heading rows may then be bolded. This type of format definition then is programmatic and changes as the values of the reports change. For example, suppose a workbook contains a report that calls for sales on a region-by-region basis. If the number of regions changes, then the report increases in the number of entries. If the region headings within a report are set to be italicized, for example, then the entry for the new region will be italicized as well. Through use of a macro, the formatting may be applied without requiring the user to modify the value for that particular row in the report and apply the italics format to that row.

Further, by including macros within the workbook, these macros may be processed by the server upon completion, thereby leveraging the processing capabilities of the server rather than using the more-scarce resources on the client-side.

The server system may also cache the workbook upon completion before transmitting the spreadsheet application to the user interface module for presentation to the user. The cached workbook may be accessed to more readily transmit information if the user happens to perform an operation on the report that calls for content that is not included in the user-provided version. For example, client-side macros may be employed that may first attempt to resolve themselves through the software application operating at the client-side. If not, however, the cached

workbook may be accessed to resolve the macro to avoid having to regenerate the reports in the workbook.

Additionally, the system provides an API to interface between the spreadsheet program macros and the report processing module. In generating a report, the components of the report are defined to be addressable features. Accordingly, through the API, various functions may be called to address the components of the report to be included in macros. This enables the user to programmatically address the data in the report. Through the use of the API, macros may be created to be included in the workbook or to be applied at the client-side upon receipt of a report. The API defines the "hooks" or metadata about the report contents that enable the macro to be able to format a report based on the contents of the report. For example, the API enables the creation of a macro that checks a report for a specific condition and then formatting the report based on the evaluation of that condition. This is "dynamic formatting" rather than the static formatting that may be performed by specifying that a specific cell is to be bold. With the API, users may define macros to be re-usable and data-independent.

For example, the following information may be provided through the API that may be used in defining macros to modify the format of the report: information about the attributes in the workbook (such as all report attributes and metrics), information about the attributes of a particular report (such as the report index number within the workbook and attributes and metrics for that report), information about the name of autoformat styles applied to the workbook or to a particular report within the workbook, information about whether autoprompt selections are applied to a report or

a workbook, information about the autoprompt selection included in the URL call, information about the number of rows in the column header of a report, information about the elements contained within the workbook or specific reports, information about the filter details within a workbook or specific report, information about the way reports are positioned on a worksheet, information about the cell range for a report, information about how a particular report is positioned within a worksheet, information about the worksheet and report name for a report within a workbook, information about the number of columns in the row header of a report, information about the URLs for specific reports on a workbook or particular report and information about whether the workbook is protected.

These types of information may be used by macros to format. For example, if a report is positioned within a report in a certain way, the macro may apply one format, but if the report is positioned in a different way, a different format may apply. The first report in a workbook may have one format and subsequent ones may have another. The possibilities therefore are provided through the provision of the API to interface with the macros.

According to one embodiment, the system comprises a report receiving object for receiving a report from the OLAP system. A report presenting object presents the report to a user using a spreadsheet application. A report displaying object then displays the report with the spreadsheet application within the network user interface. In one embodiment, the OLAP system is a relational OLAP system.

The system may also comprise an object for enabling a user to display only reports requested by the user. This provides a security feature for permitting access to

only those reports requested by a particular user. For example, a user may be denied access to display a report based on access privileges assigned to that user by a system administrator. A system administrator, however, may have privileges to access any report run against a data warehouse. Access privileges may vary from user to user.

The system may also include a personalizing object. The personalizing object permits a user to format a report based on a user's preference. For example, a user may arrange particular data in a horizontal or vertical orientation, specify a particular font or effect (*e.g.*, bold, underline) to be applied to particular portions of the report, etc. The personalizing object may use autoprompts for assigning particular properties to a report. The autoprompts may provide a user with various options relating to particular data within the report. The user may select one or more of the options (*e.g.*, using a conventional computer mouse or keyboard) to assign the properties to the report. The personalizing object may also use personalization filters.

A method for enabling users connected over a network to present reports processed by an OLAP system with a spreadsheet application within a network user interface is also disclosed. The method may include a report receiving step for receiving a report from the OLAP system. After receiving the report, the report may be presented using a report presenting step that uses the integrated spreadsheet application to present the report. A report displaying step may then be used to display the report with the spreadsheet application within the network user interface. In this manner, the user may manipulate, view, etc., the report directly within the network user interface.

According to another embodiment of the invention, a server system for presenting reports processed by an OLAP system with a spreadsheet application within a network user interface is disclosed. The server system may include a report request receiving object for receiving a report request from a user of the OLAP system. A report generating object may be used for generating a report based on the report request. A report presenting object may then present the report using the spreadsheet application within the network user interface.

Sub A2 A detailed embodiment of a system for enabling users to asynchronously generate report requests is depicted in Fig. 5. Such a system 50 comprises a data warehouse 12, server system 14, network output module 22, user systems 26, user interface modules 28 and network 36. Additionally, server system 14 may be connected to a server cache 44 and network output module 22 may be connected to a network output module cache 46. Further, a network server 42 may be provided to generate output from network output module 22 across network 36 to user systems 26.

In system 50, network 36 may comprise an intranet, the Internet or any other network system that enables communication between a server and user system such as servers 14 and 42 and user systems 26. Network server 42 may comprise a web server and user system 26 may comprise computers connected over the Internet and World Wide Web to web server 42.

Server cache 44 may be provided to store calculated report results in a relational database structure. By providing a server cache 44 with server system 14, complex business reports may be accessed with one simple server request statement rather than having to access data warehouse 12 each time that a complex business

report is requested. Additionally, through the use of server cache 44, load balancing may be provided. Further, for security purposes, server cache 44 may be used to index result tables by filter/template combination and by user.

Network output module cache 46 may be provided to store dynamic web pages created to display report results in files through network server 42. Cache results in network output module 46 may retain links to data warehouse content for *ad hoc* reporting and drilling.

Various security level architectures may be employed, although not depicted in Fig. 5. For example, a database level security may be provided between data warehouse 12 and server system 14. A network security user validation module may be provided between network output module 22 and network server 42. Further, firewall and data encryption techniques may be provided within network 36 to prevent access to network server 42 from unqualified or unauthorized user systems 26. Also, an API may be provided to enable creation of further security level clearances to prevent user systems 26 access to data and data warehouse 12 for server cache 44 to which they do not have authorization.

Operation of system 50 may now be described. System 50 may enable users to request reports from server system 14 to be processed against data warehouse 12 through the use of a user interface module, such as a web browser operatively connected over the Internet to a network output module 22. As such a report may be requested by a user through user interface module 28 of user system 26.

To access the data warehouse, a user may first log in to network server 42 via user system 26. The user may do so by selecting a particular web site maintained by

network server 42 or some other mechanism may be used. Network server 42 then presents the user with a view using user interface module 28 (*e.g.*, a web browser) comprising one or more options. A view may comprise a page or series of pages from a website, for example. One option presented in the view may be a mechanism that enables a user to generate a report request through this mechanism in the view. User interface module 28 may be used to submit a request for a report selected by the user to network server 42 over network 36. Network server 42 communicates the request to network output module 22. Network output module 22 receives the request and formulates a report request to be sent to server system 14 for processing using data warehouse 12. Network output module 22 may also perform other steps, as described in detail below with respect to Fig. 3. Network output module 22 may also cooperate with an agent module operatively connected over system 50 to formulate a report. Server system 14 then accesses data warehouse 12 to perform all data retrieval and calculations to provide the report.

Network output module 22 may provide the functionality to determine whether a report request is the same or substantially the same as a previously requested report prior to forwarding a report to server system 14 for processing. Upon receipt of a report request, network output module 22 also controls network server 42 to present a new view to the user through user interface module 28 via network 36 while continuing to monitor progress of submitted reports identified for the particular user. This enables users to perform other tasks and thereby provides the asynchronous nature of the present invention. For example, the user may operate other views, pages,

sites or any other action using user interface module 28, without having to wait for the report generation to be complete before being able to do so.

After the report generation is complete, server system 14 communicates the report to network output module 22. Network output module 22 prepares the report for presenting to the user(s) that requested the report using user interface module 28. As such, network output module 22 controls asynchronous processing of reports requested by the users.

Fig. 6 illustrates various components and objects of system 50 that perform tasks and functions described above. These objects may reside anywhere within system 50, but the location of each according to one embodiment is described as well. System 50 comprises a report requesting object 112, a report request receiving object 114, a report request submitting object 116, an option presenting object 118, a selected option receiving object 120, and a selected option submitting object 122. Report requesting object 112 communicates with report request receiving object 114 to enable a user to request a report to be generated. Report requesting object 112 may comprise functionality provided in a web page presented through user interface module 28. Report request receiving object 114 may comprise a module that is a portion of network output module 22. Report request submitting object 116 submits the user's report request for generation. Report request submitting object 116 may comprise an object that is a portion of network output module 22.

After a request has been submitted, option presenting object 118 may present various options for the report. Selected option receiving object 120 communicates with option presenting object 118 to select the option(s) selected by the user. Options

may include specification of the name, content, project, filter, template and other components of a report as may be used by server system 14. Selected option submitting object 122 then submits the options selected by the user for processing. A report generating object 124 generates the requested report asynchronously based on the options selected by the system user. A report list creating object 126 creates a report which lists one or more reports requested. If a request for a report that has already been requested, network output module 22 adds the request, including an identification of the system user requesting the report to the report entry, without generating another report. A status page creating object 128 creates a status page which identifies the status of one or more requested reports. The status page may include options which may be selected by the system user (*e.g.*, cancel request). Option presenting object 118 and selected option submitting object 122 may comprise functionality presented in a page to the user via user interface module 28. Other objects described may comprise components of network output module 22.

Through an administrator module within network output module 22, an administrator may also monitor performances of each report, filter reports by pending, error and completed requests, cancel or modify reports, and may resubmit a report if it creates an error. The administrator module may have restricted access to prevent users from canceling other user requests. The administrator module may also allow for setting global settings for the system. Further, the administrator module may also enable administrators to view history lists, status lists, and cancellation pages from the viewpoint of users on the system. This enables administrators to be able to see what

the user is seeing and therefore, troubleshoot or answer questions about the performance and operation of the system.

As described above, therefore, the present invention provides the ability for users and system administrators to cancel report requests. If a user desires to cancel a report request, the user may submit a report request cancellation by, for example, clicking on a "cancel" button using a conventional computer mouse or by selecting "cancel" from a standard pull-down menu. After a report request cancellation has been submitted, the system removes the request for that user from the report entry in the report list and then determines whether any other requests by other users remain for that report entry. If the report entry contains additional requests for that report, the system continues generating the report. This method enables users to cancel respective report requests while continuing report generation for other system users. Additionally, this reduces system overhead by generating a single report for multiple requests. For example, if several users request the same report, only one report will be generated unlike prior systems where reports are generated for each request submitted by a user. For example, if ten users submitted a single request for the same report, ten reports would be generated. The invention, however, generates only one report that is used for each of the requests submitted by the ten (10) system users.

Another feature of the invention enables notification to a system user of a completed report. Once a report is generated, the system notifies the user that requested the report via e-mail, telephone, fax, pager, or other method that the report is completed. This may be done using known broadcast technology. For example, the

invention may use Microstrategy's DSS Broadcaster™ to implement this notification feature.

Users may request and format reports generated using standard spreadsheet software using a network interface device (*e.g.*, a web browser). The web browser may be, for example, Microsoft Internet Explorer™, Netscape Navigator™, etc. Use of a web browser allows users to easily manipulate and generate without requiring that the user have a specific application loaded on the user system.

The present invention may be used in an overall system comprising a plurality of other systems for OLAP and decision support. According to one embodiment of the invention, a decision support system 10 may be provided as depicted in Fig. 7. System 10 may comprise a data warehouse 12, a server system 14, an architect module 16, an administrator module 18, a broadcast module 20, a network output module 22, a plurality of user systems 26, and an object creation module 24. Data warehouse 12 may comprise any data warehouse or data mart as is known in the art, including a relational database management system ("RDBMS") or a multidimensional database management system ("MDBMS") or a hybrid of the two. As discussed above, data warehouse 12 consolidates source system transaction data, organizes that data by business subject areas called dimensions and often includes data history for long term periods, such as two to five years. A data mart includes dimensions which tend toward application-specific subjects instead of integrating the entire business in one database such as departmental analysis data, data mining, and mass mailing lists.

Server system 14 comprises an OLAP server system for accessing and managing data stored in data warehouse 12. Server system 14 may comprise a

ROLAP engine, MOLAP engine or a HOLAP engine according to different embodiments. Specifically, server system 14 may comprise a multithreaded server for performing sophisticated analysis directly against data warehouse 12. One embodiment of server system 14 may comprise a ROLAP engine known as DSS Server™ offered by MicroStrategy.

Architect module 16 may comprise a module that enables developers to create and maintain data and metadata in data warehouse 12. Metadata may be considered to be data about data, such as data element descriptions, data type descriptions, attributes/property descriptions, range/domain descriptions, and process/method descriptions. Data and metadata stored in data warehouse 12 may thus be modified and organized by architect module 16. According to one embodiment of the invention, architect module 16 may comprise a software package known as DSS Architect™ offered by MicroStrategy.

Administrator module 18 may comprise a module for facilitating the development, deployment and management of data warehouse applications supporting large volumes of users over various distribution mechanisms. Administrator module 18 may comprise an object manager and a warehouse monitor. The object manager allows objects to be shared across databases for easy migration from development to production. The warehouse monitor provides performance monitoring and management tools to support thousands of users across a distributive database environment. The warehouse monitor collects statistics for the purpose of identifying performance bottlenecks, warehouse tuning, and cost analysis. According to one

embodiment of the invention, administrator module 18 may comprise a module known as DSS Administrator™ offered by MicroStrategy.

Server system 14 also connects to an object creation module 24. Object creation module 24 may comprise a component object model ("COM") application program interface ("API") for custom decision support development. According to one embodiment of the invention, object creation module 24 may comprise a software module known as DSS Objects™ offered by MicroStrategy. Additionally, custom applications may interface with object creation module 24 including Delphi, Visual Basic, and C++ programming modules.

Server system 14 also interfaces with a number of devices that enable users to access the contents of the data warehouse to perform various data analyses. One embodiment of such a system comprises a user system 26 with an agent module 28 operating thereon. Agent module 28 may comprise a decision support interface to enable a user to provide integrated query and reporting, powerful analytics, and decision support through any standard user system 26.

Agent module 28 may operate on any user system including personal computers, network workstations, laptop computers or any other electronic device connected to server system 14. Agent module 28 may enable the user to define queries to be performed against the data contained in data warehouse 12 using components, templates, filters, reports, agents, etc. Components may include dimensions, attributes, attribute elements, and metrics; in other words, the building blocks for templates, filters, and reports. Templates generally define a report format and specify the attributes, dimensions, metrics, and display properties comprising a

report. Filters generally qualify report content and identify the subset of data warehouse 12 to be included in a report. Reports are generally understood to be a data analysis created by combining a template (the format) with a filter (the content). Agents are a group of reports cached on a time -or event- based schedule for rapid retrieval and batch processing.

Accordingly, agent module 28 enables a user access to the contents of data warehouse 12 to provide detailed analysis on an *ad hoc* basis. According to one embodiment of the invention, agent module 28 may comprise a software package known as DSS Agent™ offered by MicroStrategy. One of the advantages of DSS Agent™ includes its use of a ROLAP architecture on server system 14 and a RDBMS in data warehouse 12 to provide a more scaleable environment. Through DSS Agent™, a user can "drill down" which allows the user to dynamically change the level of detail in a report. Drilling down allows the user to drill to a lower level attribute so that the resulting report displays data with a greater level of detail. For example, one can drill down from year to month to week to day. DSS Agent™ also enables users to "drill up" to a higher level attribute. Drilling up summarizes the selected data to a higher level total. For example, one can drill from day to week to month to year. DSS Agent™ also enables a user to "drill within." Drilling within allows a user to go to a different hierarchy within the same dimension. Drilling within is often used to examine the characteristics of selected data. For example, drilling within enables a user to drill from item to color when looking at a particular retail item such as an automobile, clothing or the like. Drilling across allows the user to drill to an altogether different dimension or subject area. For example, one can drill

across from a region to a month. Accordingly, through use of agent module 28, server system 14, and data warehouse 12, drilling is a powerful tool that is easily implemented using a ROLAP architecture which is not as easily accessible in MOLAP unless a user also implements the complicated structure of a HOLAP architecture.

Along with agent module 28, user system 26 may also include a report writing module 30, an executive module 32, and a spreadsheet module 34. Report writing module 26 may comprise an OLAP report writer. Executive module 32 may comprise a module design tool for developing custom EIS applications. This module is a design tool for developing briefing books that provide high level users with a series of views that describe their business. Once created, end users can access briefing books through agent module 28 in EIS mode. Such a system is easily implemented with agent module 28 by compiling sets of analyses into dynamic pages that immediately focus users on their key business drivers. One embodiment of executive module 32 may comprise the software made DSS Executive™ offered by MicroStrategy.

Spreadsheet module 34 may comprise an add on to existing spreadsheet programs or may comprise an entirely new spreadsheet program. Spreadsheet module 34 may enable reports and analyses generated from agent module 28 to be presented in a traditional spreadsheet program format to enable users to view results in preexisting front end interfaces. Spreadsheet module 34 may comprise the Microsoft EXCEL™ spreadsheet program offered by Microsoft and an EXCEL™ Add-In program offered by MicroStrategy.

Another module for accessing content of server system 14 may comprise a network output module 22. Network output module 22 offers the advantage of

enabling user system 26 access to server system 14 in data warehouse 12 without requiring an additional agent module 28 to be stored on user system 26. Instead, user system 26 may have a user interface module 38 residing on user system 26. User interface module 38 may comprise any module that enables a user system to interface with the network output module over a network 36. According to one embodiment of the invention, network 36 may comprise an intranet or the Internet or other developed Internet-type networks. Further, user interface module 38 may comprise any standard browser module such as Microsoft Internet Explorer™ or Netscape Navigator™. As most user systems 26 already have a user interface module 38 stored and operating thereon, network output module 22 offers the advantage of enabling users access to server system 14 in data warehouse 12 without learning to operate a new module such as agent module 28. One embodiment of network output module 22 may comprise a web-based module called DSS Web™ offered by MicroStrategy. Accordingly, in one embodiment, a user can access server system 14 through a standard web browser, such as Microsoft Internet Explorer™, over the Internet through network output module 22, such as DSS Web™.

In this embodiment, network output module 22 may comprise a World Wide Web tool used in conjunction with server system 14 for allowing users to deploy data warehouse/decision support applications over the Internet using industry standard World Wide Web browsers as a client. As a result, a user can access the data warehouse with little or no client maintenance, little or no software to install, and only a small amount of additional training while still maintaining all of the capabilities of agent module 28. One embodiment of such a network output module 22 comprises

DSS Web™ offered by MicroStrategy. Such an embodiment provides a broad array of options for viewing information sets, such as spreadsheet grids and a wide variety of graphs. Through this module's exceptional reporting capabilities, users receive key elements of a report in easily interpretable, plain messages. This module also allows users to "drill" dynamically to a lower level of detail to view the underlying information or to create and save new analyses. For sensitive information, this module provides security plug-ins that allow the user to extend the standard security functionality with additional user authentication routines. This module may also provide an interface API that allows users to customize, integrate, and imbed this functionality into other applications. For example, the data syndicator for health care information could utilize this module with a customize interface to sell access to health care information to Health Maintenance Organizations, hospitals, and pharmacies.

Another module that enables user access to the data warehouse 12 comprises a broadcast module 20. Broadcast module 20 may comprise an information broadcast server designed to deliver personalized messages to multiple recipients via user devices 40. User devices 40 may comprise electronic mail, facsimile, pager, mobile phone, telephone, and multiple other types of user information devices. Broadcast module 20, like agent module 28 and network output module 22, enables users to define queries and reports that are to be run against server system 14 and data warehouse 12. Broadcast module 20 may then send personalized information to users at predefined intervals or when criteria specified in their reports exceed predefined thresholds. Broadcast module 20 may continually monitor and run such reports to

enable users to receive appropriate information as it becomes available. Broadcast module 20 may also personalize the report to ensure that users see only that portion of a report that is relative to that user and thereby frees analysts from acting as the information provider. Broadcast module 20 may also provide a wide range of content formatting options to enable users to receive information in a form that they can best understand. One such formatting may include formatting a request for a pager, facsimile, mobile phone, electronic mail, web or other output user device 40.

Broadcast module 20 may specify a mechanism called a service for generating information and a subscription that determines which users receive information from which service. Accordingly, a service may be tied to, for example, a specific report, series of reports or elements within a report, etc. Also, subscribers may include users, groups of users or only specific user devices 40 for a particular user. Broadcast module 20 may output information to user devices 40 using push technology. In other words, as soon as a criteria is established, broadcast module 20 generates a report to a user device 40. Accordingly, a user can thus have up-to-date information about the contents of data warehouse 12 without having to submit a query or log-in to a software module such as agent module 28 or network output module 22 to initiate the query.

In such an overall system, the present invention may enable asynchronous report generation through operation of network output module system 22 as described above. Other embodiments and uses of the invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention

disclosed herein. The specification and examples should be considered exemplary only. The scope of the invention is only limited by the claims appended hereto.